Short Answer:

Answer the following questions with complete sentences in your own words. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers. You are expected to be able to explain your answers in detail (Provide examples for each question).

1. What are objects in C# and why do we need objects?
2. Explain the ***static*** keyword in C#. What are static fields and static methods?
3. What are extension methods? How to use extension methods? Benefits of extension methods.
4. What is Inheritance and how many types of inheritance are supported by C#?
5. What is the diamond problem? And how can we resolve the problem?
6. What is an interface and what is an abstract class? What are the differences between them?
7. What is Polymorphism?
8. What are the differences between overriding and overloading?
9. What is Encapsulation? How does C# implement it? And why do we need encapsulation?
10. What does { get; set; } mean in C#?
11. What is the difference between abstraction and encapsulation?
12. What is the ***sealed*** keyword and when do we need it?
13. Can we use ***this*** keyword in the constructor to invoke other constructors? Why?
14. What is the difference between ***virtual*** keyword and ***abstract*** keyword?
15. What is the Object class in C#? Can you list and explain some methods in the Object class?
16. What are Boxing and Unboxing?
17. What's the rule for object casting?

Coding Questions:

Write code in c# to solve the following problems. Please write your own answers. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it is easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. Write an extension method for string type, to change all the letters in the calling string to upper cases.

2. This is a credit card number validation problem. Credit card numbers follow certain patterns:
   - A credit card number must have between 13 and 16 digits.
   - It must start with:
     - 4 for Visa cards
     - 5 for Master cards
     - 37 for American Express cards
     - 6 for Discover cards

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card number is entered correctly or whether a credit card is scanned correctly by a scanner.

Credit card numbers are generated following this validity check, commonly known as the Luhn check or the Mod 10 check, which can be described as follows (for illustration, consider the card number 4388576018402626):

Step 1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number.
4388576018402626
2 * 2 = 4
2 * 2 = 4
4 * 2 = 8
1 * 2 = 2
6 * 2 = 12 (1 + 2 =3)
5 * 2 = 10 (1 + 0 =1)
8 * 2 = 16 (1 + 6 =7)
4 * 2 = 8

Step 2. Now add all single-digit numbers from Step1.
4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37

Step 3. Add all digits in the odd places from right to left in the card number
6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38

Step 4. Sum the results from Step 2 and Step 3
37 + 38 = 75

Step 5. If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is

invalid, but the number 4388576018410707 is valid.

Your task is to write a program that prompts the user to enter a credit card number as a long integer. Display whether the number is valid or invalid.

Design your program to use the following methods:

```
/** Return true if the card number is valid */
public static bool isValid(string number)
/** Get the result from Step B */
public static int sumOfDoubleEvenPlace(string number)
/** Return this number if it is a single digit, otherwise, return the sum of the two digits */
public static int getDigit(string number)
/** Return sum of odd-place digits in number */
public static int sumOfOddPlace(string number)
/** Return true if the digit d is a prefix for number */
public static bool prefixMatched(string number, int d)
/** Return the number of digits in d */
public static int getSize(long d)
/** Return the first k number of digits from number. If the number of digits in number is less than k, return the number. */
public static long getPrefix(string number, int k)
```

3. Design a class named **Person** and its two subclasses named **Student** and **Employee**. Make **Faculty** and **Staff** subclasses of **Employee**.
   - A Person has a name, address, phone number, and email address.
   - A Student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. (Here it means to define class status as an enum class.)
   - An Employee has an office, salary, and date hired.
   - A Faculty member has office hours and a rank.
   - A Staff member has a title.

You should also override the ToString() method of the above classes to display the class name and the person's name.

Your task is to write a test program that creates a Person, Student, Employee, Faculty, and Staff, and invokes their ToString() methods.

4. Guessing Game Project

   A. Develop a simple class that represents a number guessing game. The game is played by the program randomly generating a number and the user attempting to guess that number. After each guess, the program will provide a hint to the user identifying the relationship between the number and the guess. If the guess is above the answer then "Too High" is returned, if the guess is below the answer then "Too Low". Also if the difference between the answer and the guess is less than the difference between the answer and the previous guess, "Getting warmer" is returned. If the difference between the answer and the guess is more

than the difference between the answer and the previous guess, then “Getting Colder” is returned.

B. The program will allow the user to play multiple games. Once a game is complete the user will be prompted to play a new game or quit.

C. Design and build a **GuessingGame** class.

- 7 instance variables -
  - *answer* - an integer representing the randomly generated number.
  - *generator* – a random Generator object
  - *gameOver* – a Boolean, false if the game is still in progress, true if the game is over.
  - *differential* – an integer representing the difference between a guess and the answer.
  - *max* – maximum value of the number to guess. For example, if the maximum number is 100 then the number to guess would be between 0 and 100. (inclusive)
  - *maxGuessesAllowed* – the maximum number of guesses the user gets, once this value is passed the game is over.
  - *numGuessesTaken* – an integer that stores the number of guesses taken so far in any game.
- Constructor and methods -
  - Default Constructor
    - Sets max to zero.
    - Creates the random number generator object.
  - Parameterized Constructor
    - Takes an integer parameter representing the maximum value of the number to guess.
    - Creates the random number generator object.
  - NewGame() method
    - Takes in an integer as a parameter representing the maximum number of guesses and sets the value for *maxGuessesAllowed*.
    - Generates the answer using the random number generator. (0 - *max*).
    - Sets *gameOver* to false.
    - Sets *differential* to the max value.
    - Sets *numGuessTaken* to zero.
  - Guess() method
    - Takes in an integer as a parameter representing a new guess.
    - Compares the new guess with the answer and generates and returns a String representing an appropriate response.
    - The response is based on:
      - The relation of the guess and answer (too high, too low, or correct).
      - The comparison of the difference between the current guess and the answer and the previous guess and the answer. (warmer, colder)

- Guess out of range error, if the guess is not between 0 and the max number (inclusive) (see sample run below)
- User has taken too many guesses because *numGuessesTaken* is greater than *maxGuessesAllowed*. If this is the case, set *gameOver* to true.

- IsGameOver() method - returns the state of game.
  - true if game is over.
  - false if still in progress.

• Accessor and mutator methods for all instance fields except the Random number generator. Use the Accessor or mutator methods within the other methods of the class rather than directly accessing the instance fields. For example, use mutator methods in the parameterized constructor to modify instance variables.

D. Design and build **GuessingGameTester** class

• This program will create **GuessingGame** objects and completely test the GuessingGame Class.

• Hint: The tester shall also provide a nested loop - one loop allows the user to play a new game after the previous game is completed; another loop will prompt the user for a new guess and provide a response based on the guess.

Example -

Sample Output – (Note: text in orange represents the user input value)

```
Welcome to the Guessing Game
Enter the maximum number:
100
Enter the number of guess allowed:
6
Enter your guess, remember it must be between 0 and 100.
50
Too High
Getting Warmer
Enter your guess, remember it must be between 0 and 100.
25
Too High
Getting Warmer
Enter your guess, remember it must be between 0 and 100.
12
Too High
Getting Warmer
Enter your guess, remember it must be between 0 and 100.
6
```

```
Too low
Getting Warmer
Enter your guess, remember it must be between 0 and 100.
8

Congratulation
Would you like to play again, enter Y for yes, N for no.
Y

Welcome to the Guessing Game
Enter the maximum number:
50
Enter the number of guess allowed:
5
Enter your guess, remember it must be between 0 and 50.
60
Guess out of range, The guess must be between 0 and 50.
Enter your guess, remember it must be between 0 and 50.
25
Too low
Getting Warmer
Enter your guess, remember it must be between 0 and 50.
48
Too High
Getting Colder
Enter your guess, remember it must be between 0 and 50.
37
Too High
Getting Warmer
Enter your guess, remember it must be between 0 and 50.
36
Congratulation
Would you like to play again, enter Y for yes, N for no.
N
```